APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: SCALING IMAGES FOR DISPLAY

APPLICANT: LOUIS A. LIPPINCOTT

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV399292282US

April 20, 2004

Date of Deposit

SCALING IMAGES FOR DISPLAY

BACKGROUND

This invention relates to scaling images for display.

In video images, the number of pixels in an image determines the quality, or resolution, of the image. More pixels in an image translates to higher resolution.

High definition television (HDTV) images, for example, have a high resolution (e.g., 1920x540 1080i (interlaced)) that cannot be directly displayed on a typical personal computer (PC) monitor without scaling the images to a lower resolution (e.g., 1280x720). Additionally, the images typically occupy a small section of the PC monitor, requiring further scaling to a lower resolution (e.g., 480x135).

One way to scale HDTV images to a usable size for PC monitors is by overlay scaling. Overlay scaling reads an HDTV image from the computer's memory and scales it horizontally and vertically. Overlay scales images "on the fly," while the PC monitor is being refreshed. The scaled image replaces ("overlays") the previously scaled image being displayed on the PC monitor. The number of images scaled and displayed per second, e.g., 85 frames per second (85Hz), enables a computer

user to view a continuous video picture sequence on the monitor.

For example, in FIG. 1, overlay scaling reads a 1920x540 1080i HDTV image 12 from a PC's memory, creates a 4:1 downscaled 480x135 image 14, and displays the image 14 on the PC's monitor. As seen in FIG. 2, image 12 is one of a sequence of incoming video images that appear at an image update rate of 60 frames per second (60Hz) and are stored temporarily in memory 13. Because image 12 is interlaced (only every other line of the image is displayed), the "real" update rate is 30 frames per second (30Hz). The overlay process reads successive images 12 from computer memory 13 at a PC CRT (cathode-ray tube) refresh rate, e.g., 85 frames per second (85Hz), downscales them, and delivers them to the monitor for display.

Also referring to FIG. 3, to create the 4:1 downscaled image 14, an overlay process reads sixteen pixels of image 12 (one pixel segment 16), compresses them to form one pixel of image 14, displays the one pixel, and proceeds to the next segment 16. The segments 16 are processed from left to right in each row, working from the top row to the bottom row. This overlay scaling requires an average memory bandwidth of 176MB/sec, where memory bandwidth equals (horizontal resolution, 1920) x (vertical resolution, 540) x (refresh rate, 85) x (bytes per pixel, 2), and a peak memory bandwidth

of 1054MB/sec (1920x540x85x12). Some PC memory systems cannot supply such a high bandwidth, so the PC compensates by dropping lines of the image. For example, dropping every other line would reduce the bandwidth requirements by 50%. Dropping lines, however, decreases image quality because the information in every pixel should contribute to the downscaled image.

SUMMARY

In general, in one aspect, the invention features scaling a graphic image that has pixels arranged in rows and columns by processing a succession of segments. Each segment comprises contiguous pixels. The row and column dimensions of each segment do not correspond to an intended degree of scaling in both dimensions. The processing of each segment produces an intermediate pixel. The intermediate pixels form a stream. The intermediate stream of pixels is processed to form a final two-dimensional scaled image.

In another aspect, the invention features scaling each image that appears in a video sequence of images for display on a display device that displays downscaled images by compressing each image in a first scaling process to form a sequence of intermediate, partially scaled images, and

compressing each of the intermediate images in a second scaling process to form a final sequence of scaled images.

Other advantages and features will become apparent from the following description and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram illustrating overlay scaling.

FIG. 2 is a diagram showing video image processing.

FIG. 3 is a diagram of pixels.

FIG. 4 is a diagram illustrating a two-pass scaling technique.

FIG. 5 is a diagram of pixels.

FIG. 6 is a diagram of pixels.

FIG. 7 is a block diagram of a two-pass scaling technique.

FIG. 8 is a block diagram of a vertical scaling process.

FIG. 9 is a table of inputs to an input pixel formatting block.

FIG. 10 is a table of outputs from an input pixel formatting block.

FIG. 11 is a diagram of input and output pins on an input pixel formatting block.

FIG. 12 is a table of inputs to a pixel filtering block.

FIG. 13 is a table of outputs from a pixel filtering block.

FIG. 14 is a diagram of input and output pins on a pixel filtering block.

FIG. 15 is a block diagram of a pixel filtering block.

FIG. 16 is a table of inputs to an output pixel formatting block.

FIG. 17 is a table of outputs from an output pixel formatting block.

FIG. 18 is a diagram of input and output pins on an output pixel formatting block.

FIG. 19 is a block diagram of a horizontal scaling process.

DESCRIPTION

In a specific example shown in FIG. 4, a two-pass scaling technique vertically and horizontally scales a 1920x540 1080i HDTV image 22 stored in memory. A first pass 18 vertically scales the image 22 to a 4:1 vertically scaled 1920x135 image 24. A second pass 20 horizontally scales vertically scaled image 24 to a 4:1 vertically and horizontally scaled 480x135 image 26, the final image (ignoring any deinterlacing processing).

Also referring to FIG. 2, image 22 is one of a sequence of incoming video images that are received at an image update rate, e.g., 60 frames per second (60Hz), by an HDTV receiving system. Each image 22 in the sequence is downscaled, requiring frequent access to memory 13 (where each image 22 is stored). The total amount of memory that a PC can read in a given period is called memory bandwidth. Passes 18 and 20 use a total memory bandwidth of 199 MB/sec and a peak bandwidth of 264 MB/sec, fitting the capabilities of the typical PC. This peak memory bandwidth is less than the 1054MB/sec peak memory bandwidth in the FIG. 1 overlay scaling example which exceeds the typical PC's capabilities.

In the first pass 18, image 22 is read from memory 13 at an image update rate, e.g., 60 frames per second (60Hz). As shown in FIG. 5, to create a vertically scaled image 24, the first pass 18 reads four memory locations to fetch four horizontal pixel segments 25 (four pixels of image 22). Each horizontal pixel segment 25 includes 32 bits (four pixels of eight bits each) and travels on a 32-bit data bus to a four-tap filter where they await compression. The horizontal pixel segments 25 may vary in size, e.g., eight pixels of eight bits each, to fit on a different sized bus, e.g., 64 bits. With the four horizontal pixel segments 25 stored in the four-tap filter, the first pass 18 compresses four vertical pixel segments 29 (formed from horizontal pixel segments 25) at the

image update rate to form four pixels of vertically scaled image 24. The first pass 18 then proceeds to process the remaining horizontal pixel segments 25 from top to bottom, working from the left column to the right column.

This reading and compressing uses a sustained memory bandwidth of 124MB/sec, where memory bandwidth equals (horizontal resolution, 1920) x (vertical resolution, 540) x (refresh rate, 60Hz) x (bytes per pixel, 2). The first pass 18 stores vertically scaled image 24 in memory 13, which uses a sustained memory bandwidth of 31MB/sec (1920x135x85x2). Thus, the first pass 18 uses a total sustained memory bandwidth of 155MB/sec (124 MB/sec + 31MB/sec).

A second pass 20 horizontally scales vertically scaled image 24 to create final image 26 using overlay scaling. In the second pass 20, vertically scaled image 24 is read from memory 13 at a PC CRT refresh rate, e.g., 85 frames per second (85Hz). As shown in FIG. 6, to create final image 26, the second pass reads a horizontal pixel segment 27 of four pixels of vertically scaled image 24, compresses them at the PC CRT refresh rate to form one pixel of final image 26, and proceeds to the next horizontal segment 27. The horizontal segments 27 are processed from top to bottom, working from the left column to the right column.

The second pass uses an average memory bandwidth of 44MB/sec (1920x135x85x2) and a peak memory bandwidth of 264

MB/sec (1920x135x85x12). Adding the average memory bandwidths for passes 18 and 20 produces the total memory bandwidth used in the two-pass scaling technique, 199 MB/sec (155MB/sec + 44MB/sec).

Thus, as shown in FIG. 7, in the two-pass scaling technique, an image is stored in memory 50. In a vertical scaling process 48, a vertical scaling function 52 reads the image from memory 50 and scales it vertically. The vertically scaled image is stored back in memory 53. In a horizontal scaling process 58, the second scaling pass, a horizontal scaling function 54 reads the vertically scaled image from memory 53 and scales it horizontally. The result of the horizontal scaling process 58 is displayed on a PC's display screen 56.

In an example of a structure implementing a vertical scaling process, shown in FIG. 8, an image to be scaled is stored in a memory 28. A memory interface 30 enables other blocks to read and write in memory 28. A hardware vertical scaler 40 vertically scales the image using blocks 32-36.

An input pixel formatting block (IPFB) 32 requests horizontal pixel segments of the image 22 from memory interface 30, generates the addresses required to gather the horizontal pixel segments in the proper sequence, formats them to the expected form, and sends them to the second block, a pixel filtering block (PFB) 34. The PFB 34 filters the

vertical pixel segments formed from the horizontal pixel segments as it receives them from the IPFB 32. This filtering effects the vertical scaling, thus making the PFB 34 the key block in the hardware vertical scaler 40. After filtering the pixels, the PFB 34 outputs them to the third block, an output pixel formatting block (OPFB) 36. The OPFB 36 receives pixels from the PFB 34, collects them for rendering back to memory interface 30, and generates the addresses required to assemble the vertically scaled image 24 in memory 30.

The first block in the hardware vertical scaler 40 is the input pixel formatting block (IPFB) 32, implemented as an integrated circuit having inputs as shown in FIG. 9 and outputs as shown in FIG. 10.

The IPFB 32 starts its operations when it receives a start command(s). As seen in FIG. 11, the IPFB 32 starts when it sees an Off to On transition on at least one of the inputs Start_Y_Cmnd 60 (command to start the processing of the Y (luminance) plane of the pixels), Start_U_Cmnd 62 (command to start the processing of the U (red minus luminance) plane of the pixels), and Start_V_Cmnd 64 (command to start the processing of the V (blue minus luminance) plane of the pixels). Processing occurs in the order of Y, U, then V, so processing begins on the first plane with an active start command. Once active, the start command(s) 60-64 must not be removed until a Done signal 74 is activated.

The start command(s) 60-64 causes the IPFB 32 to start fetching pixels from memory 28 through the memory interface 30, handshaking with the memory interface 30 with fetch inputs and outputs 78, starting at the location indicated by an x_Addr 66 (current address value), where "x" represents Y, U, or V, whichever matches the start command currently being processed. The memory 28 and memory interface 30 are configured so that reading a single memory location fetches, in this example, four contiguous pixels.

Also referring to FIG. 5, four horizontal pixel segments 25 of four pixels of image 22 are read from top to bottom, working from the left column to the right column. Concurrently, a value at an x_Pitch 68, representing the horizontal size of the horizontal pixel segment 25, is added to the x_Addr 66, indicating the address of the next horizontal pixel segment 25. Horizontal pixel segments 25 are so read in columns until the value in an x_Length 70 is met, indicating the end of a column.

After reading a column, the IPFB 32 asserts a Column_Done 76 signal to the PFB 34, and the IPFB 32 resets the x_Addr 66 to point to the top horizontal pixel segment 25 in the next column. The pixel reading so continues until the value in an x_Width 72 is met, indicating that all columns have been read. Once met, an x_Done 74 becomes active, indicating the end of reading the x plane. If its start command is active, the next

plane in the sequence (U or V) is processed in the same way. Once all three x_Done 74 signals are active, the IPFB 32 ceases operation until the next start command at any Start_x_Cmnd 60-64.

Therefore, in general what the IPFB 32 does is retrieve horizontal pixel segments 25 (four pixels for a 4:1 downscaling) of image 22 from memory and present them to the PFB 34 for vertical scaling.

The second block in the hardware vertical scaler 40 is the PFB 34, implemented as an integrated circuit with inputs as shown in FIG. 12 and outputs as shown in FIG. 13.

As shown in FIGS. 14 and 15, the PFB 34 starts its operations when it sees an Off to On transition, triggered by x_Done 74, on one of a Y_Done 80, U_Done 82, or V_Done 84 input. A filter datapath 35 in the PFB 34 fetches two horizontal pixel segments 25 (one quad of data) at a time from the IPFB 32. The horizontal pixel segments 25 are fetched from top to bottom, working from the left column to the right column. As the PFB 34 reads horizontal pixel segments 25, it filters them based on a scale factor. The PFB 34 can properly operate on both luminance and chrominance (subsampled) pixels. The PFB 34 outputs the number of filtered pixels to form the same sized pixel segment as the PFB received to an output formatting block (OPFB) 36.

Therefore, in general what the PFB 34 does is vertically scale pixel segments (four pixels for a 4:1 downscaling) of the original image 22 and output the scaled pixels to the OPFB 36.

The third block in the hardware vertical scaler 40 is the OPFB 36, implemented as an integrated circuit with inputs as shown in FIG. 16 and outputs as shown in FIG. 17.

As shown in FIG. 18, the OPFB 36 starts its operations when it sees an Off to On transition, triggered by the Store_x 94 from the PFB 34, on a corresponding Store_Y 100, Store_U 102, or Store_V 104 input. The OPFB 36 uses scaled inputs and outputs 114 to handshake with the PFB 34 and receive the vertically scaled pixel segments to be rendered in memory interface 30. The OPFB 36 resets an x_Addr 106 (current address value) to point to the storage location for the first vertically scaled pixel segment 27 (see FIG. 6) it receives from the PFB 34. Since the PFB 34 does not perform any horizontal resizing, input image 22 and output image 24 have the same horizontal dimension, allowing for just one set of registers describing the image width and pitch values for the IPFB 32 and the OPFB 36.

The OPFB 36 buffers the vertically scaled pixel segments 27 in a first in, first out (FIFO) queue. From the FIFO queue, the OPFB 36 stores each vertically scaled pixel segment 27 in the memory interface 30, handshaking with the memory

interface 30 using store inputs and outputs 116. The memory interface 30 stores the vertically scaled pixel segments 27 in memory 28.

Concurrent with buffering the pixel segment 27, the OPFB 36 adds the value at an x_Pitch 108, representing the size of the scaled pixel segment 27, to the appropriate x_Addr 106, indicating the address of the next vertically scaled pixel segment 27. Vertically scaled pixel segments 27 are so buffered until the value in an x_Length 110 is met, indicating the end of a column.

At the end of reading a column, during the transfer of the bottom vertically scaled pixel segment 27 (the flushing of the FIFO queue) to the memory interface 30, the OPFB 36 resets the x_Addr 106 to point to the top pixel segment 27 in the next column. The pixel buffering so continues until the value in an x_Width 112 is met, indicating that all columns have been buffered. At this point, the image has been vertically scaled and rendered to memory interface 30 by the hardware vertical scaler 40.

Therefore, in general what the OPFB 36 does is receive vertically scaled pixel segments from the PFB 34 and render them to memory interface 30.

Turning to the block diagram in FIG. 19, in a horizontal scaling process, a polyphase filter 122 reads the vertically scaled image 24 one horizontal pixel segment at a time from

memory 120. The polyphase filter 122 horizontally scales each pixel segment as it receives it. The polyphase filter 126 reads four pixels of vertically scaled image 24 and compresses them to form one pixel of final image 26.

As shown in FIG. 6, and discussed above as overlay scaling, the pixels in vertically scaled image 24 are read four pixels (one horizontal pixel segment 27) at a time from top to bottom, working from the left column to the right column. The vertically and horizontally scaled image 26 replaces ("overlays") the previously scaled image 26 being displayed on the PC monitor 124. The number of images displayed per second at the PC CRT rate, e.g., 85 frames per second (85Hz), enables a computer user to view a continuous video picture sequence on the monitor 124. The screen rate will repeat some images every second in order to approximate the slower image update rate, e.g., 60 frames per second (60Hz).

Other embodiments are within the scope of the following claims.